

Identity Management System

(Fundamental Java Project)



Submitted by-

Vaaruni Joshi (MSc-SE)

Ritu Shikha (MSc-SE)

TABLE OF CONTENTS

1. SUBJECT DESCRIPTION	#
2. SUBJECT ANALYSIS	#
2.1 MAJOR FEATURES	#
2.2 APPLICATION FEASIBILITY	#
2.3 DATA DESCRIPTION	
2.4 EXPECTED RESULTS	
2.5 SCOPE OF THE APPLICATION	#
3. CONCEPTION	#
3.1 CHOSEN ALGORITHM	#
3.2 DATA STRUCTURES	
3.3 GLOBAL APPLICATION FLOW	
3.4 GLOBAL SCHEMA AND MAJOR FEATURES SCHEMA	#
4. CONSOLE OPERATIONS DESCRIPTION	#
5. CONFIGURATION INSTRUCTIONS	#
6. COMMENTED SCREENSHOTS	#
7. BIBLIOGRAPHY	#

1. Subject Description

The project “**IAMCore**” is an Identity Management Application which is developed to manage and control the users information in a secured way. The objective of this application is to manage the User Identities – categorized into Admin, User.

2. Subject Analysis

2.1 Major features

It's a console application where the user can perform the following function on various identities stored in the database

- Create an identity
- Update an identity
- Delete an identity
- List the identities
- Search the identities

For the security of the information of the identities we have enforced access control by implementing user authentication where the user has to login first using an ID/username and a password to perform the above listed functions.

2.2 Application Feasibility

The development of this application with the given features is quite feasible. The application has a simple database architecture; it is password secured which makes it authentication dependent that means the data is secured.

2.3 Data Description

Below we have defined the types of data we created to make this application achieve its given features:

- Class Identity

We have defined the features of identity in this class and methods like getters and setters for these features also defined in this class.

Variables

- int uid
- string displayName
- string email
- string password
- boolean isAdmin

Methods

- identity()
- boolean isAdmin()
- void setAdmin()
- string getUid()
- void setUid()
- string getDisplayName()
- void setDisplayName()
- string getEmail()
- void setEmail()
- string getPassword()
- void getPassword()
- string toString()

- Interface DAO

This interface defines the outline of all the functions that have to be performed on the database.

Methods

- public void create(T entity)
- public void update(T entity)
- public void delete(T entity)
- public T find(final Object id)
- List<T> search(String[] keywords, T entity)

- IdentityDAO

The class IdentityDAO implements the interface DAO. This class defines all the functions that are performed on the database such as creating, updating, deleting, searching and viewing the identities.

Methods

- public IdentityDao
- public void create(Identity entity)
- public void update(Identity entity)
- public void delete(Identity entity)
- public List <Identity> readAll()
- public Identity find(Object id)

- `public List<Identity> search(String[] keywords, Identity entity)`

2.4 Expected result

The application is expected to authenticate the user through ID/Username and Password. After that, user should be able to perform below functions:

- **Create** an Identity
- **Update** an Identity
- **Delete** an Identity
- **List** All Identities
- **Search** the Identities

2.5 Scope of the application

Limitations:

- It's only a console application with no GUI.
- Only the user that is the admin can perform the primary functions such as delete/modify/search the identities
- UID cannot be modified.
- Deletion is done for only one identity at a time.
- The data of identities is not that detailed and only has 3 fields.

Evolutions:

- GUI can be developed for the application.
- Multiple entries can be deleted at a time
- The other users which are not admins can have the option to modify their own information
- It can be enhanced to store more detailed data about identities making it usable for other identity management systems such as university database of students/teachers, library management, bank management etc

3. Conception

3.1 Data structures

The data structure used in this project is ArrayList, Boolean,String
In java an array is a static data structure where as an array list supports dynamic arrays meaning that the size is not fixed an it can grow or shrink automatically if something is added or deleted from the ArrayList.

IDENTITY_ID	DISPLAY_NAME	EMAIL	PASSWORD	ISADMIN

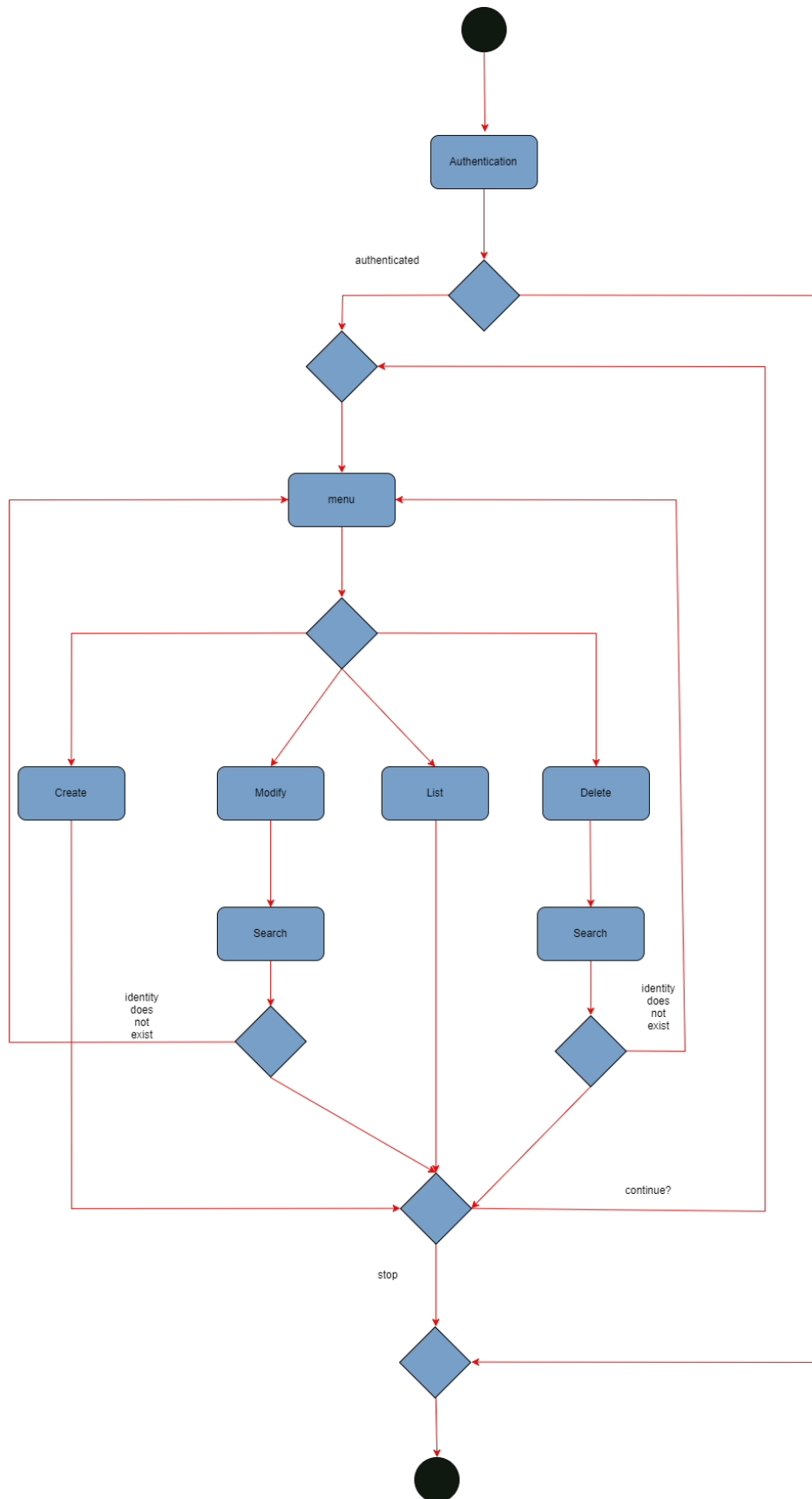
3.2 [Chosen algorithm](#)

The application uses Java ArrayList for storing or accessing Identity or configuring data. Also, application handles the identity data through Identity Class.

Data Access Object Pattern or DAO pattern is used to separate low level data accessing API or operations from high level business services.

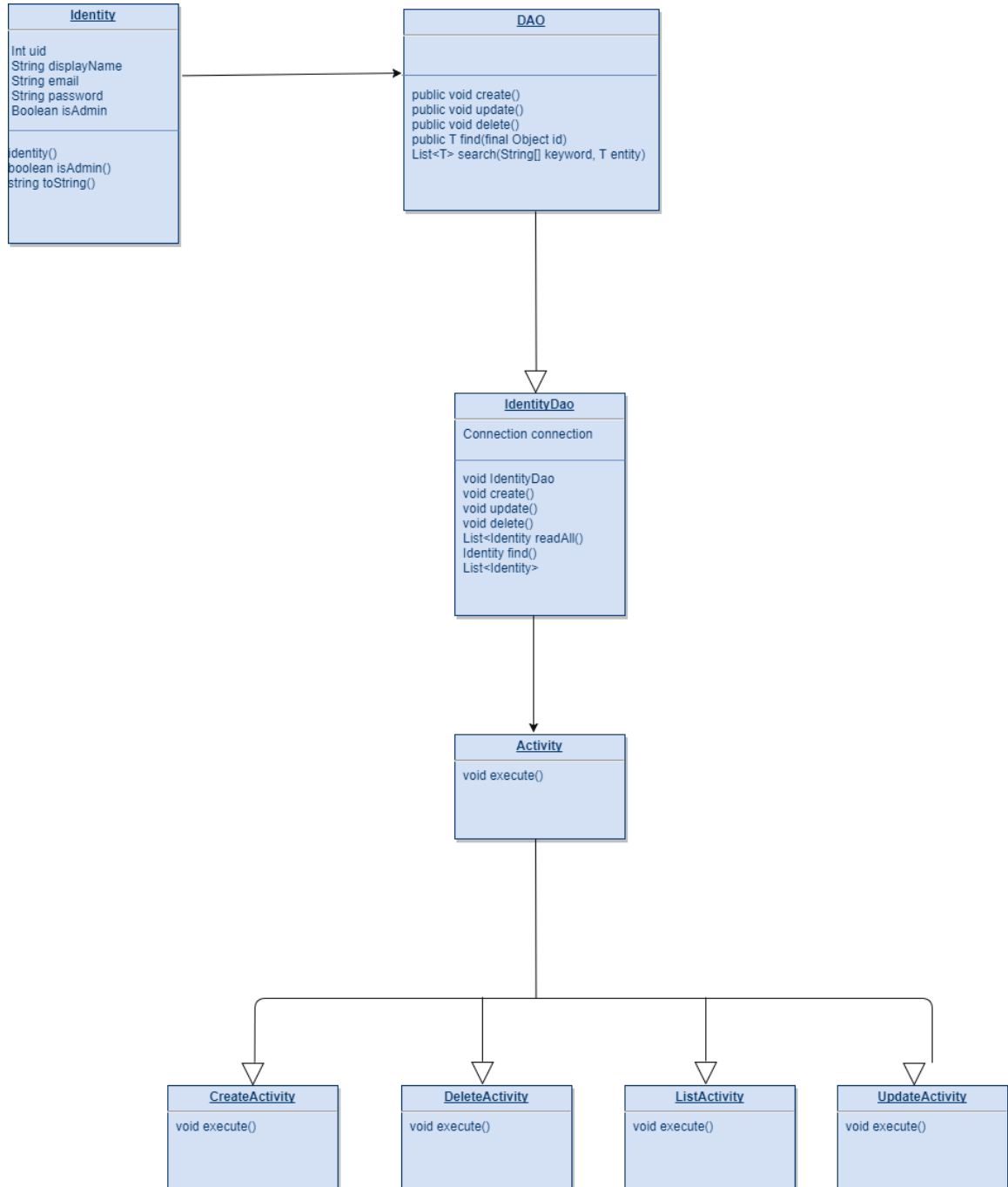
Data can be persisted by reading/writing data from/in a database using a JDBC connector.

3.3 [Global application flow](#)

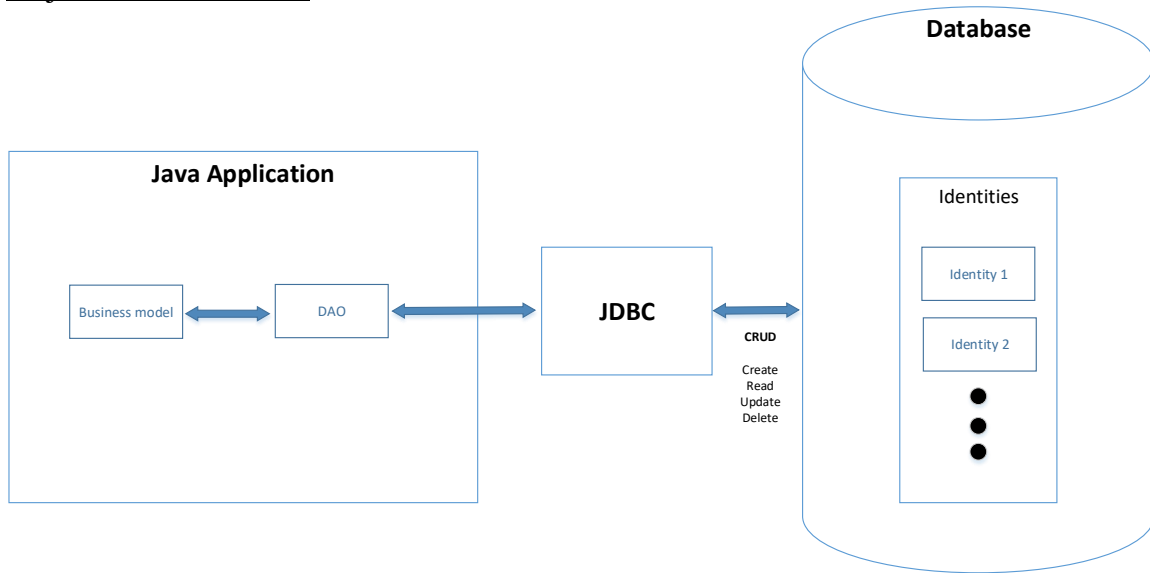


3.4 Global schema and major features schema

Global schema



Major Features Schema



4. Console Operations Description

The identity management application is a console based application, the user has to select corresponding operations for input from the console and the output will also be displayed on the console.

Authentication: User will be prompted to login as to authenticate him and has to submit his username and password.

Menu: After authentication a menu will be displayed and the following functionalities can be performed:

Menu

Please select an action:

1. Create an identity
2. Modify an identity
3. Delete an identity
4. List an identity
5. Quit

Option 1: If user selects 1, he will be asked to enter all the details for creation of the identity like
Display name, email and password one by one.

Option 2: If user selects 2, he will be asked to enter the identity id which he wants to modify and after that the user will have to select the field he wants to update. After entering the new data the application will again display a menu

asking if the user wants to update another field, if not the user can select “update” which will modify the details. After that the application will display the modified version of the data.

Option 3: If user selects 3, he will be asked to enter the identity id which he wants to delete

Option 4: If user selects 4, all the identities stored in the database will be displayed

Option 5: Choosing 5 will exit the application.

5. Configuration Instruction

This application was built using

Platform: Platform Independent

IDE: Eclipse Latest version: Oxygen 4.7.1a

JDK: JDK 1.8

DataBase: Derby DBMS

Steps to run the Identity management application

1. Setup the database:

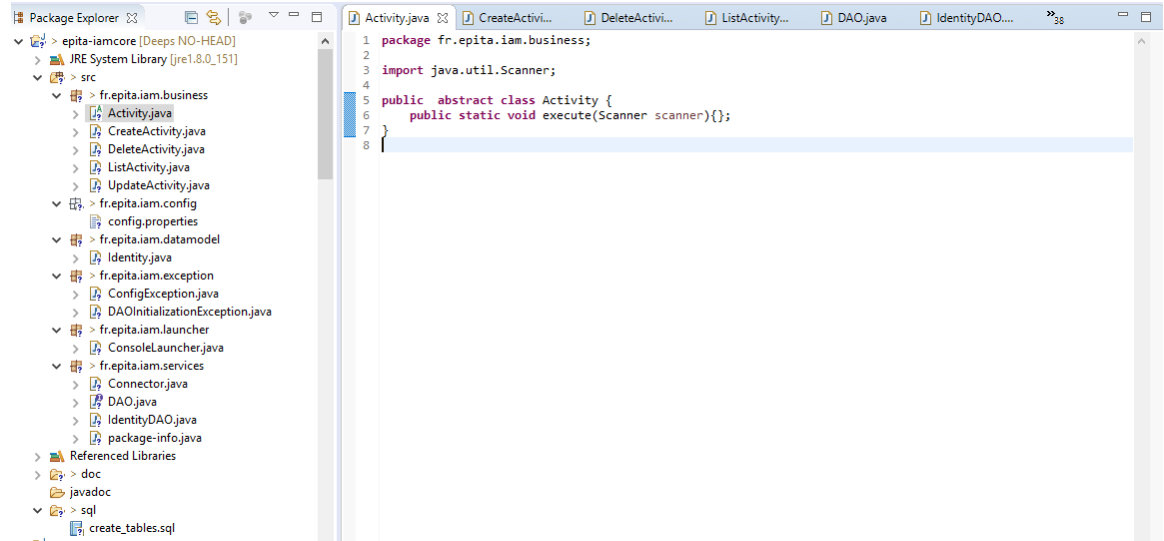
- Download apache derby client
- Create an instance of derby name it instance name =IAM with username="admin" and password="admin".
- run /db-derby-10.13.1.1-bin/bin/startNetworkServer
- Go to iam-core/src/fr/epita/iam/config/config.properties edit user value with the user name you used to create your derby schema Do the same with password, the name if instance if you used your own values
- Go to iam-core/sql run create_tables.sql on your instance IAM

2. Run the application:

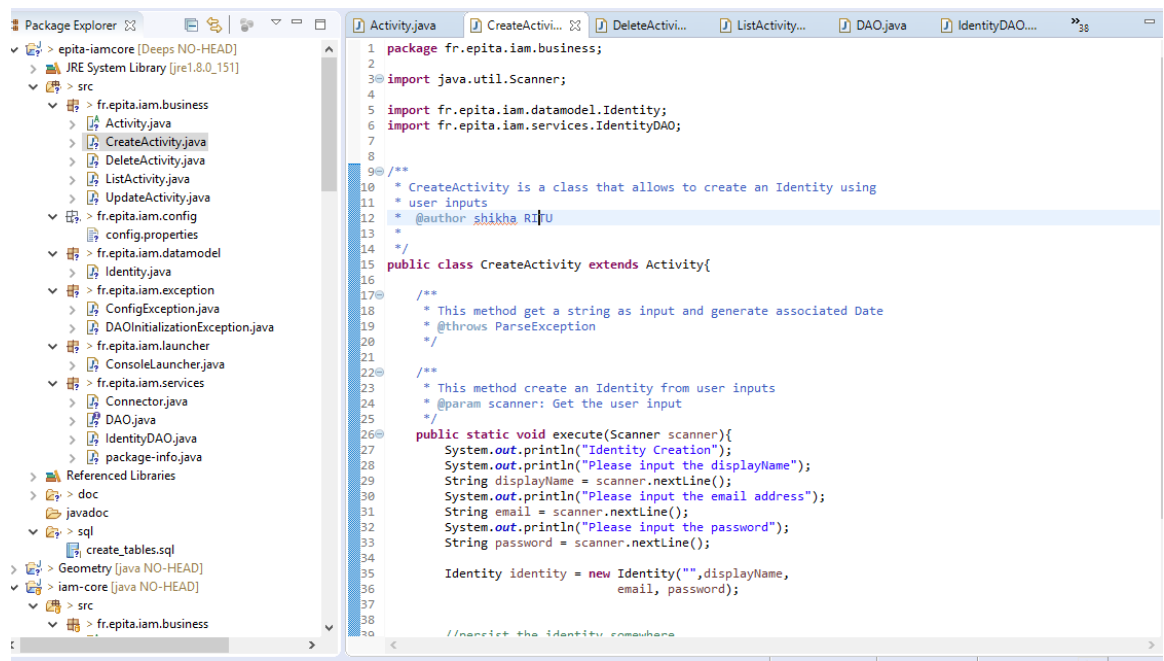
- Go to iam-core/src/fr/epita/iam/launcher execute ConsoleLauncher.java

6. Commented Screenshots

Activity.java



CreateActivity.java



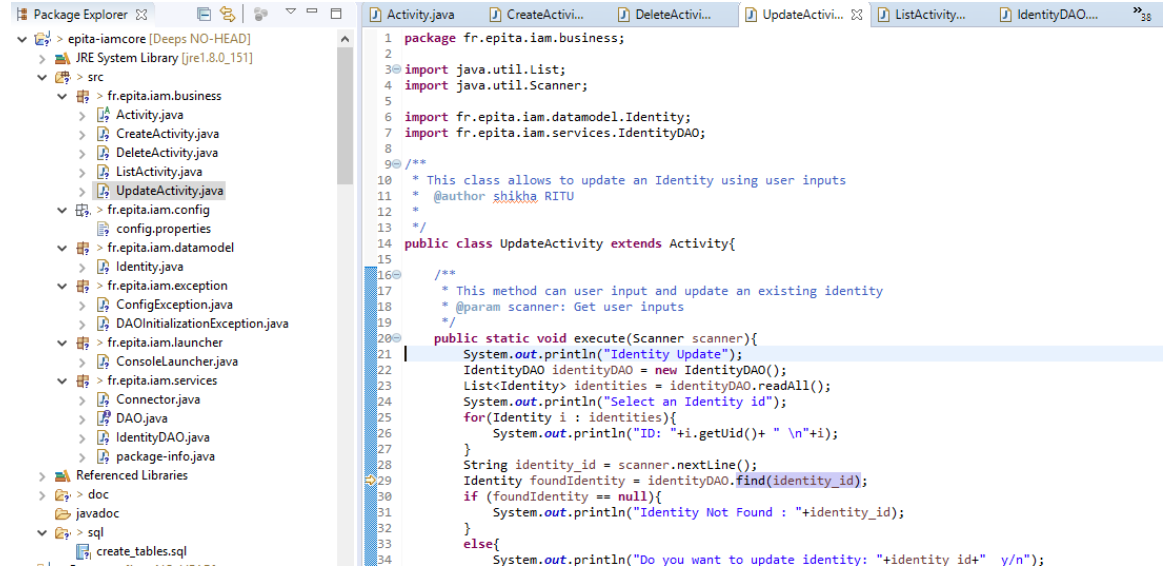
DeleteActivity.java

```
1 package fr.epita.iam.business;
2
3
4
5 import java.util.List;
6 import java.util.Scanner;
7 import fr.epita.iam.datamodel.Identity;
8 import fr.epita.iam.services.IdentityDAO;
9
10 /**
11  * This class allows to delete an Identity selected by the user
12  * @author shikha RITU
13  */
14
15 public class DeleteActivity extends Activity {
16
17     /**
18      * This method delete an existing Identity
19      * @param scanner: Get user selection
20      */
21     public static void execute(Scanner scanner){
22         System.out.println("Identity Deleting");
23         IdentityDAO identityDAO = new IdentityDAO();
24         List<Identity> identities = identityDAO.readAll();
25
26         System.out.println("Select an identity id");
27         for(Identity i : identities){
28             System.out.println("ID : "+i.getId()+ " \n"+i);
29         }
30         String identity_id = scanner.nextLine();
31         if (identity_id.isEmpty()) {
32             System.out.println("Did not understand you answer ");
33             return;
34         }
35         Identity foundIdentity = identityDAO.find(identity_id);
36         if (foundIdentity == null){
37             System.out.println("Identity Not Found : "+identity_id);
38         }
39     }
40 }
```

ListActivity.java

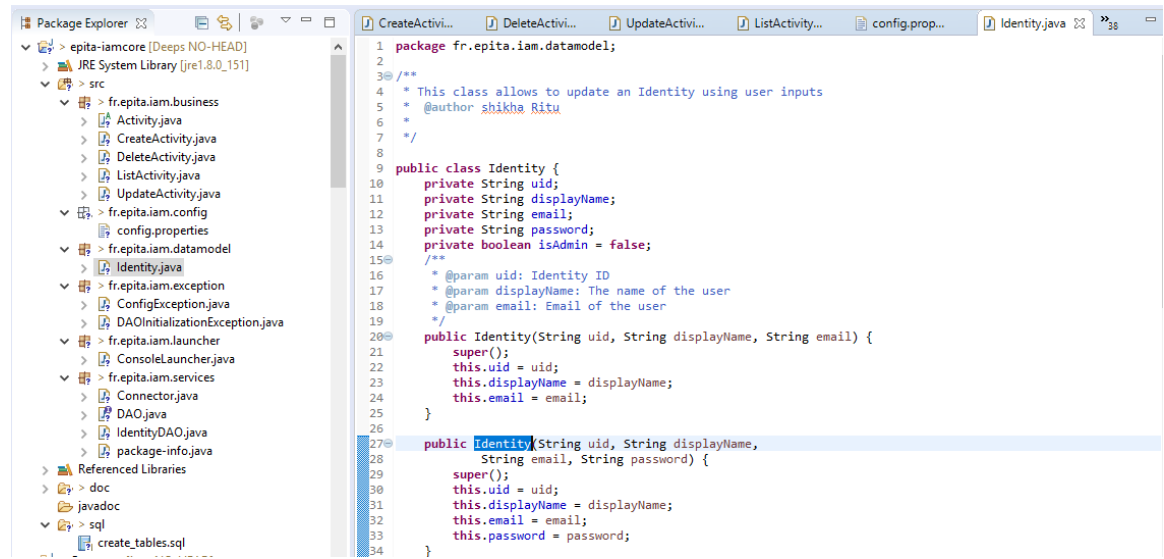
```
1 package fr.epita.iam.business;
2
3 import java.util.List;
4
5 /**
6  * This class allows to delete an Identity selected by the user
7  * @author shikha RITU
8  */
9
10 /**
11  * ListActivity is a class that allows to Get list of an Identity using
12  * user inputs
13  * @author shikha RITU
14  */
15
16 public class ListActivity extends Activity{
17
18     public static void execute(){
19         System.out.println("Identity Listing");
20         IdentityDAO identityDAO = new IdentityDAO();
21         List<Identity> identities = identityDAO.readAll();
22         for(int i = 0; i < identities.size(); i++){
23             System.out.println( i+ " ." + identities.get(i).toString());
24         }
25     }
26 }
27
28 }
```

UpdateActivity.java



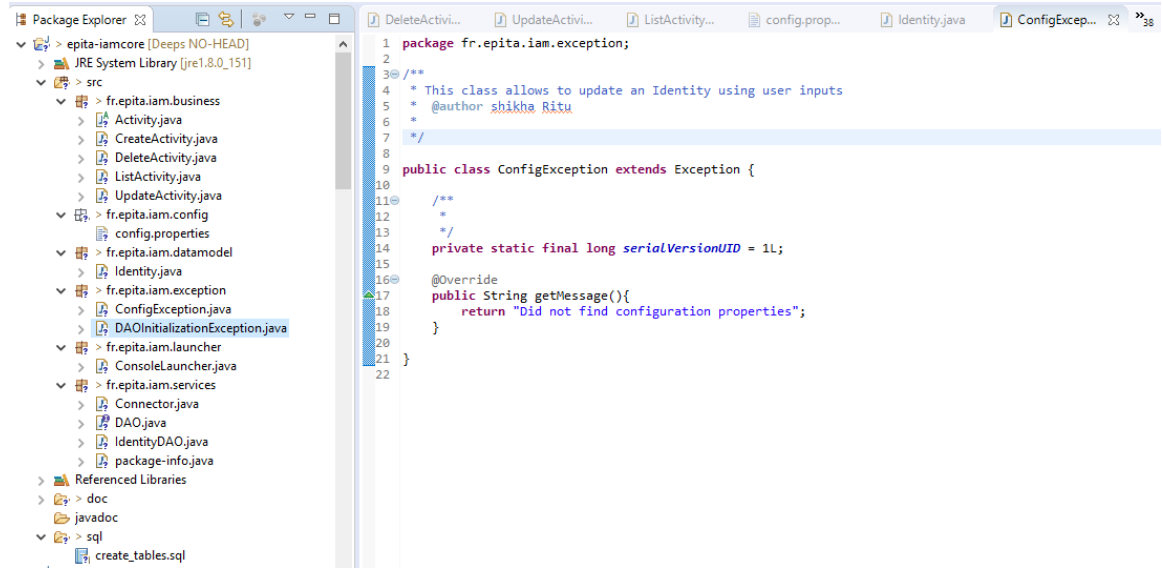
```
1 package fr.epita.iam.business;
2
3 import java.util.List;
4 import java.util.Scanner;
5
6 import fr.epita.iam.datamodel.Identity;
7 import fr.epita.iam.services.IdentityDAO;
8
9 /**
10  * This class allows to update an Identity using user inputs
11  * @author shikha RITU
12  */
13
14 public class UpdateActivity extends Activity{
15
16     /**
17      * This method can user input and update an existing identity
18      * @param scanner: Get user inputs
19      */
20     public static void execute(Scanner scanner){
21         System.out.println("Identity Update");
22         IdentityDAO identityDAO = new IdentityDAO();
23         List<Identity> identities = identityDAO.readAll();
24         System.out.println("Select an Identity id");
25         for(Identity i : identities){
26             System.out.println("ID: "+i.getId()+ " \n"+i);
27         }
28         String identity_id = scanner.nextLine();
29         Identity foundIdentity = identityDAO.find(identity_id);
30         if (foundIdentity == null){
31             System.out.println("Identity Not Found : "+identity_id);
32         }
33         else{
34             System.out.println("Do you want to update identity: "+identity_id+" y/n");
```

Identity.java



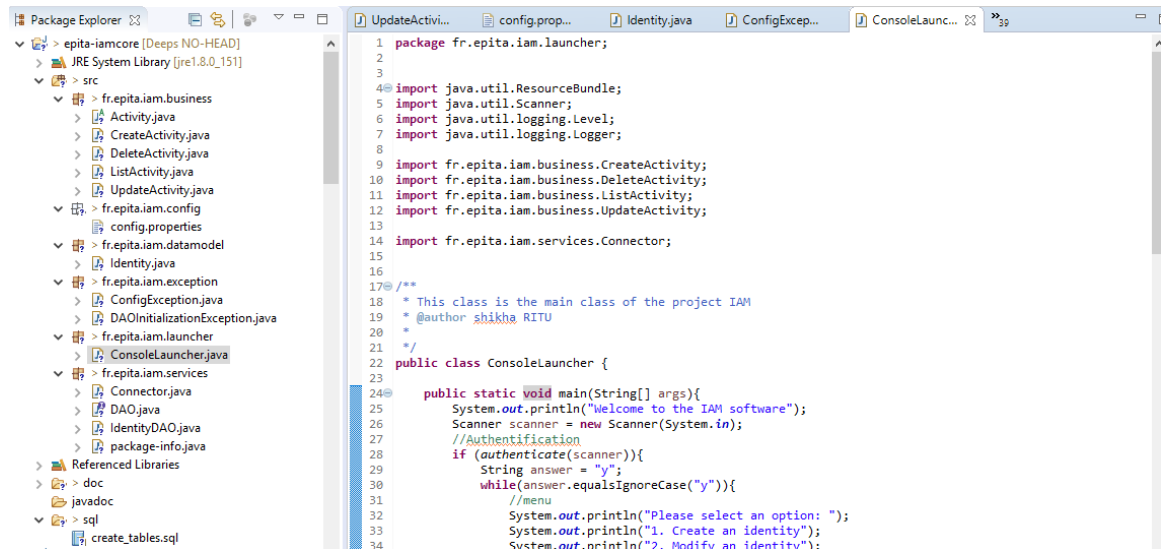
```
1 package fr.epita.iam.datamodel;
2
3 /**
4  * This class allows to update an Identity using user inputs
5  * @author shikha Ritu
6  */
7
8
9 public class Identity {
10     private String uid;
11     private String displayName;
12     private String email;
13     private String password;
14     private boolean isAdmin = false;
15
16     /**
17      * @param uid: Identity ID
18      * @param displayName: The name of the user
19      * @param email: Email of the user
20      */
21     public Identity(String uid, String displayName, String email) {
22         super();
23         this.uid = uid;
24         this.displayName = displayName;
25         this.email = email;
26     }
27
28     public Identity(String uid, String displayName,
29                     String email, String password) {
30         super();
31         this.uid = uid;
32         this.displayName = displayName;
33         this.email = email;
34         this.password = password;
35     }
36 }
```

ConfigException.java



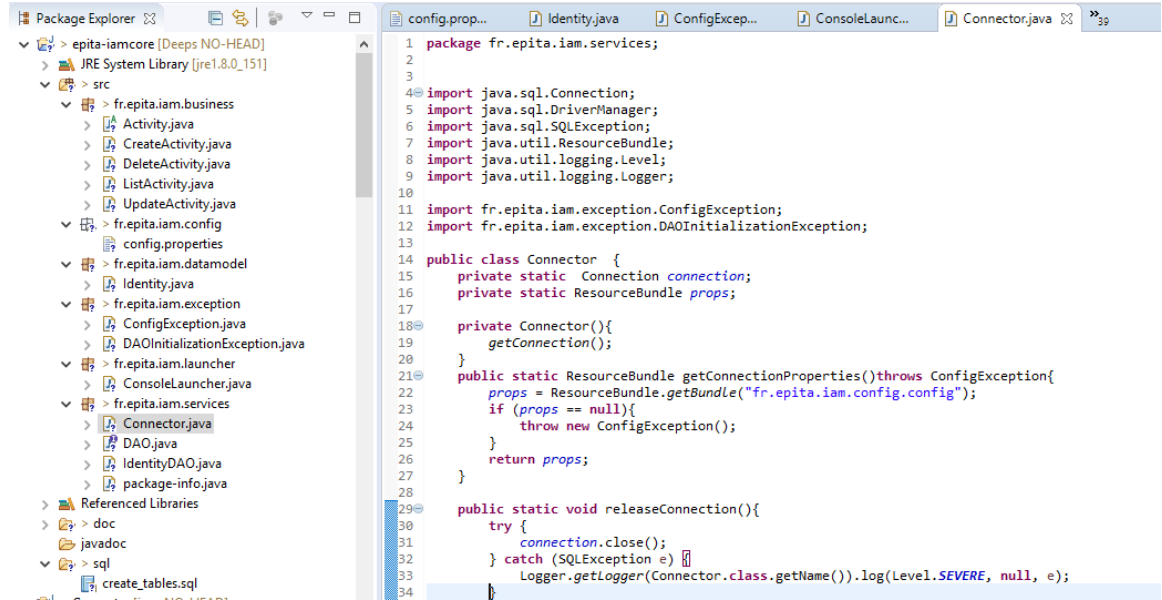
```
1 package fr.epita.iam.exception;
2
3 /**
4  * This class allows to update an Identity using user inputs
5  * @author shikha Ritu
6  */
7
8
9 public class ConfigException extends Exception {
10
11     /**
12      *
13      */
14     private static final long serialVersionUID = 1L;
15
16     @Override
17     public String getMessage(){
18         return "Did not find configuration properties";
19     }
20
21 }
```

ConsoleLauncher.java



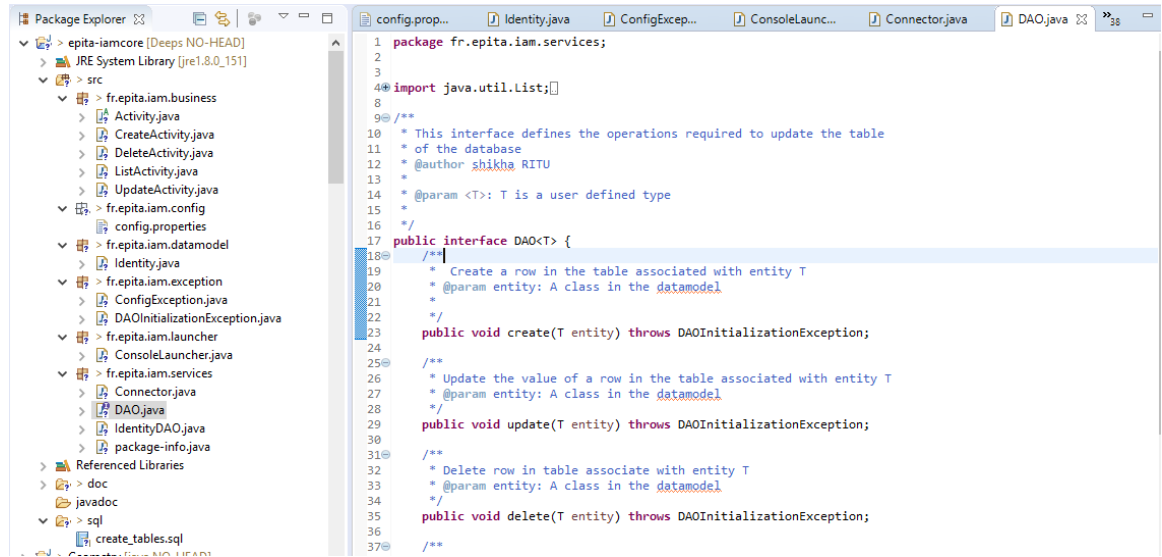
```
1 package fr.epita.iam.launcher;
2
3
4 import java.util.ResourceBundle;
5 import java.util.Scanner;
6 import java.util.logging.Level;
7 import java.util.logging.Logger;
8
9 import fr.epita.iam.business.CreateActivity;
10 import fr.epita.iam.business.DeleteActivity;
11 import fr.epita.iam.business.ListActivity;
12 import fr.epita.iam.business.UpdateActivity;
13
14 import fr.epita.iam.services.Connector;
15
16
17 /**
18  * This class is the main class of the project IAM
19  * @author shikha RITU
20  */
21
22 public class ConsoleLauncher {
23
24     public static void main(String[] args){
25         System.out.println("Welcome to the IAM software");
26         Scanner scanner = new Scanner(System.in);
27         //Authentication
28         if (authenticate(scanner)){
29             String answer = "y";
30             while(answer.equalsIgnoreCase("y")){
31                 //menu
32                 System.out.println("Please select an option: ");
33                 System.out.println("1. Create an identity");
34                 System.out.println("2. Modify an identity");
35             }
36         }
37     }
38 }
```

Connector.java



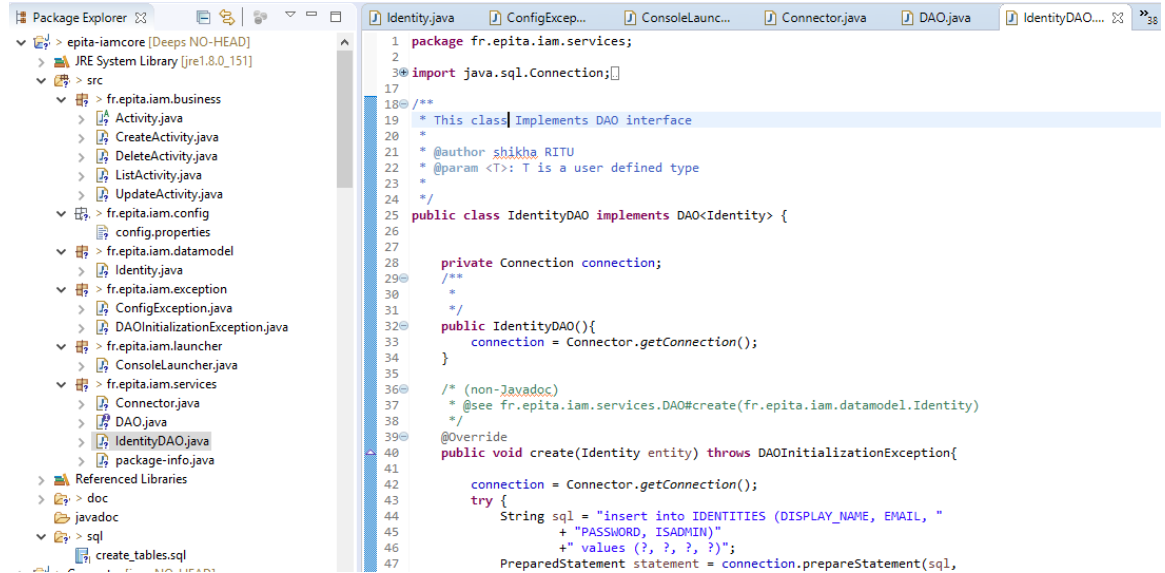
```
1 package fr.epita.iam.services;
2
3
4 import java.sql.Connection;
5 import java.sql.DriverManager;
6 import java.sql.SQLException;
7 import java.util.ResourceBundle;
8 import java.util.logging.Level;
9 import java.util.logging.Logger;
10
11 import fr.epita.iam.exception.ConfigException;
12 import fr.epita.iam.exception.DAOInitializationException;
13
14 public class Connector {
15     private static Connection connection;
16     private static ResourceBundle props;
17
18     private Connector(){
19         getConnection();
20     }
21
22     public static ResourceBundle getConnectionProperties()throws ConfigException{
23         props = ResourceBundle.getBundle("fr.epita.iam.config.config");
24         if (props == null){
25             throw new ConfigException();
26         }
27         return props;
28     }
29
30     public static void releaseConnection(){
31         try {
32             connection.close();
33         } catch (SQLException e) {
34             Logger.getLogger(Connector.class.getName()).log(Level.SEVERE, null, e);
35         }
36     }
37 }
```

DAO.java



```
1 package fr.epita.iam.services;
2
3
4 import java.util.List;
5
6 /**
7  * This interface defines the operations required to update the table
8  * of the database
9  * @author shikha RITU
10  *
11  * @param <T>: T is a user defined type
12  */
13
14 public interface DAO<T> {
15
16     /**
17      * Create a row in the table associated with entity T
18      * @param entity: A class in the datamodel
19      */
20     public void create(T entity) throws DAOInitializationException;
21
22     /**
23      * Update the value of a row in the table associated with entity T
24      * @param entity: A class in the datamodel
25      */
26     public void update(T entity) throws DAOInitializationException;
27
28     /**
29      * Delete row in table associate with entity T
30      * @param entity: A class in the datamodel
31      */
32     public void delete(T entity) throws DAOInitializationException;
33
34 }
```

IdentityDAO.java



```
1 package fr.epita.iam.services;
2
3 import java.sql.Connection;[]
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18 /**
19  * This class implements DAO interface
20  *
21  * @author shikha RITU
22  * @param <T>: T is a user defined type
23  */
24
25 public class IdentityDAO implements DAO<Identity> {
26
27
28     private Connection connection;
29     /**
30      *
31      */
32     public IdentityDAO(){
33         connection = Connector.getConnection();
34     }
35
36     /* (non-Javadoc)
37      * @see fr.epita.iam.services.DAO#create(fr.epita.iam.datamodel.Identity)
38      */
39     @Override
40     public void create(Identity entity) throws DAOInitializationException{
41
42         connection = Connector.getConnection();
43         try {
44             String sql = "insert into IDENTITIES (DISPLAY_NAME, EMAIL, "
45                 + "PASSWORD, ISADMIN)"
46                 + " values (?, ?, ?, ?)";
47             PreparedStatement statement = connection.prepareStatement(sql,
```

7. Bibliography

- [1. http://www.javatpoint.com/java-tutorial](http://www.javatpoint.com/java-tutorial)
- [2. http://thomas-broussard.fr](http://thomas-broussard.fr)